DOCUMENT RESUME

ED 261 655 IR 011 804

AUTHOR Hutchins, Edwin L.; And Others
TITLE Direct Manipulation Interfaces.
INSTITUTION California Univ., San Diego, La Jolla. Inst. for Cognitive Science.
SPONS AGENCY Office of Naval Research, Arlington, Va.
REPORT NO ICS-8503
PUB DATE May 85
CONTRACT N00014-85-C-0133; NR-667-541
NOTE 35p.; For related document, see IR 011 805.
PUB TYPE Information Analyses (070) -- Viewpoints (120)

EDRS PRICE MF01/PC02 Plus Postage.
DESCRIPTORS Cognitive Development; *Cognitive Processes; Computers; *Computer Software; *Data Processing; *Man Machine Systems; Programing Languages
IDENTIFIERS *Cognitive Psychology; *Iconic Representation

ABSTRACT

This paper presents a cognitive account of both the advantages and disadvantages of direct manipulation interfaces, i.e., the use of icons to manipulate and interact directly with data rather than writing programs or calling on a set of statistical subroutines. Two underlying phenomena that give rise to the sensation of directness are identified. One is concerned with the information processing distance between the user's intentions and the facilities provided by the machine; the second is concerned with the relation between the input and output vocabularies of the interface language. It is noted that direct manipulation requires that the system provide representations of the objects of interest that behave as if they were the objects themselves in order to create the sensation of directness of manipulation. Concluding remarks admit the complexity of understanding direct manipulation interfaces, but stress that a properly designed system should be easy for the user to learn, enabling people to spend their time learning the task domain rather than the computer system. Lists of 20 references and 30 reports by people in the Institute for Cognitive Science complete the document. (JB)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED261655

U.S. DEPARTMENT OF EDUCATION
NATIONAL INSTITUTE OF EDUCATION
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)
☒ This document has been reproduced as received from the person or organization originating it.
☐ Minor changes have been made to improve reproduction quality.
• Points of view or opinions stated in this document do not necessarily represent official NIE position or policy.

BEST COPY AVAILABLE

DIRECT MANIPULATION INTERFACES

Edwin L. Hutchins, James D. Hollan, and Donald A. Norman

May 1985

ICS Report 8503

BEST COPY AVAILABLE

DIRECT MANIPULATION INTERFACES

Edwin L. Hutchins, James D. Hollan,
and Donald A. Norman

May 1985

ICS Report 8503

NPRDC-UCSD Intelligent Systems Group
Institute for Cognitive Science
University of California, San Diego C-015
La Jolla, CA 92093

To be published in D. A. Norman & S. W. Draper (Eds.) (1986), *User centered system design: New perspectives in human-computer interaction.* Hillsdale, NJ: Lawrence Erlbaum Associates.

The research reported here was conducted under Contract N00014-85-C-0133, NR 667-541 with the Personnel and Training Research Programs of the Office of Naval Research. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the sponsoring agency. Approved for public release; distribution unlimited. Reproduction in whole or in part is permitted for any purpose of the United Stated Government. Requests for reprints should be sent to Edwin L. Hutchins, Institute for Cognitive Science, C-015; University of California, San Diego; La Jolla, CA 92093.

Copyright © 1985 by Edwin L. Hutchins, James D. Hollan, and Donald A. Norman

IR011804

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) ICS 8503	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Institute for Cognitive Science 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) C-015 University of California, San Diego La Jolla, CA 92093	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Personnel & Training Research 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N-00014-85-C-0133
8c. ADDRESS (City, State, and ZIP Code) Code (442 PT) Office of Naval Research 800 North Quincy Street, Arlington, VA 22217	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. NR 667-541 / TASK NO. / WORK UNIT ACCESSION NO

11. TITLE (Include Security Classification)
Direct Manipulation Interfaces

12. PERSONAL AUTHOR(S)
Edwin L. Hutchins, James D. Hollan, & Donald A. Norman

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM 12/84 TO 5/85	May 1985	26

16. SUPPLEMENTARY NOTATION
To be published in D. A. Norman & S. W. Draper (Eds.), User centered system design: New perspectives in human-computer interaction. Hillsdale, NJ: Lawrence Erlbaum Assoc.

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Human-machine interface; cognition; direct manipulation; interface language;

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Direct manipulation has been lauded as a good thing in interface design, and some interfaces that have this property have been well received by users. In this paper we seek a cognitive account of both the advantages and disadvantages of direct manipulation interfaces. We identify two underlying phenomena that give rise to the sensation of directness. One has to do with the information processing distance between the user's intentions and the facilities provided by the machine. Reduction of this distance makes the interface feel direct by reducing the effort required of the user to accomplish goals. The second phenomenon concerns the relation between the input and output vocabularies of the interface language. In particular, direct manipulation requires that the system provide representations of the objects of interest that behave as if they are the objects themselves. This provides the sensation of directness of manipulation.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code) / 22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
unclassified

Contents

DIRECT MANIPULATION: ITS NATURE, FORM, AND HISTORY 1
A Brief History of Direct Manipulation 3
The Goal: A Cognitive Account of Direct Manipulation 4
TWO ASPECTS OF DIRECTNESS: DISTANCE AND ENGAGEMENT 5
Distance 5
Direct Engagement 7
TWO FORMS OF DISTANCE: SEMANTIC AND ARTICULATORY 9
Semantic Directness 9
Semantic Distance in the Gulfs of Execution and Evaluation 10
The Gulf of Execution. 10
The Gulf of Evaluation. 10
Reducing the Semantic Distance That Must Be Spanned 11
Higher-level languages. 11
Make the output show semantic concepts directly. 12
Automated behavior does not reduce semantic distance. 13
The user can adapt to the system representation. 14
Virtuosity and semantic distance. 14
Articulatory Directness 15
Articulatory Distance in the Gulfs of Execution and Evaluation 16
Iconographic languages. 18
DIRECT ENGAGEMENT 18
A SPACE OF INTERFACES 19
PROBLEMS WITH DIRECT MANIPULATION 21
Reassessing the Claims for Direct Manipulation 21
CONCLUDING REMARKS 24
ACKNOWLEDGMENTS 25
REFERENCES 25

Direct Manipulation Interfaces

EDWIN L. HUTCHINS, JAMES D. HOLLAN, and DONALD A. NORMAN

DIRECT MANIPULATION: ITS NATURE, FORM, AND HISTORY

The best way to describe a *Direct Manipulation* interface is by example. Suppose we have a set of data to be analyzed with the numbers stored in matrix form. Their source and meaning are not important for this example: The numbers could be the output of a spreadsheet, a matrix of numerical values from the computations of a conventional programming language, or the results of an experiment. Our goal is to analyze the numbers, to see what relations exist among the rows and columns of the matrix. The matrix of numbers is represented on a computer display screen by an icon. To plot one column against another, we simply get a copy of a graph icon, then draw one line from the output of one column to the x-axis of the graph icon and another line from the output of the second column to the y-axis input of the graph icon: See Figure 1A. Not what was wanted? Erase the lines and reconnect them. Want to see other graphs? Make more copies of the graph icons and connect them. Need a logarithmic transformation of one of the axes? Move up a function icon, type in the algebraic function that is desired ($y = \log x$, in this case) and connect it in the desired data stream. Want the mean and standard deviation of one of the columns? Connect the column to the appropriate statistical icon. These examples are illustrated in Figure 1B.

Now consider how we could partition the data. Suppose one result of our analysis was the scatter diagram shown in Figure 2A. The straight line that has been fitted through the points is clearly inappropriate. The data fall into two quite different clusters and it would be best to analyze those clusters separately. In the actual data matrix, the points that form the two clusters might be scattered randomly throughout the data set. The regularities are apparent only when we plot them. How do we pull out the clusters? Suppose we could simply circle the points of interest and use each circled set as if it were a new matrix of values, each of which could be analyzed in standard ways, as shown in Figure 2B.

The examples of Figures 1 and 2 are partially implemented. partially hypothetical: But they illustrate a powerful manipulation medium for computation.[1] The promise of Direct Manipulation is that instead of an abstract computational medium, all the 'programming' is done graphically, in a form that matches the way one thinks about the problem. The desired operations are done simply by moving the appropriate icons onto the screen and connecting them together. Connecting the icons is the equivalent of writing a program or calling on a set of statistical subroutines, but with the advantage of being able to directly manipulate and interact with the data and the connections. There are no hidden operations, no syntax or command names to learn. What you see is what you get. Some classes of syntax errors are eliminated. For example, you can't point at a non-existent object. The system requires expertise in the

[1] The examples are part of a design specification for our research project and at the time this chapter was written, part of the system had been implemented, including most of what is shown in Figure 1.

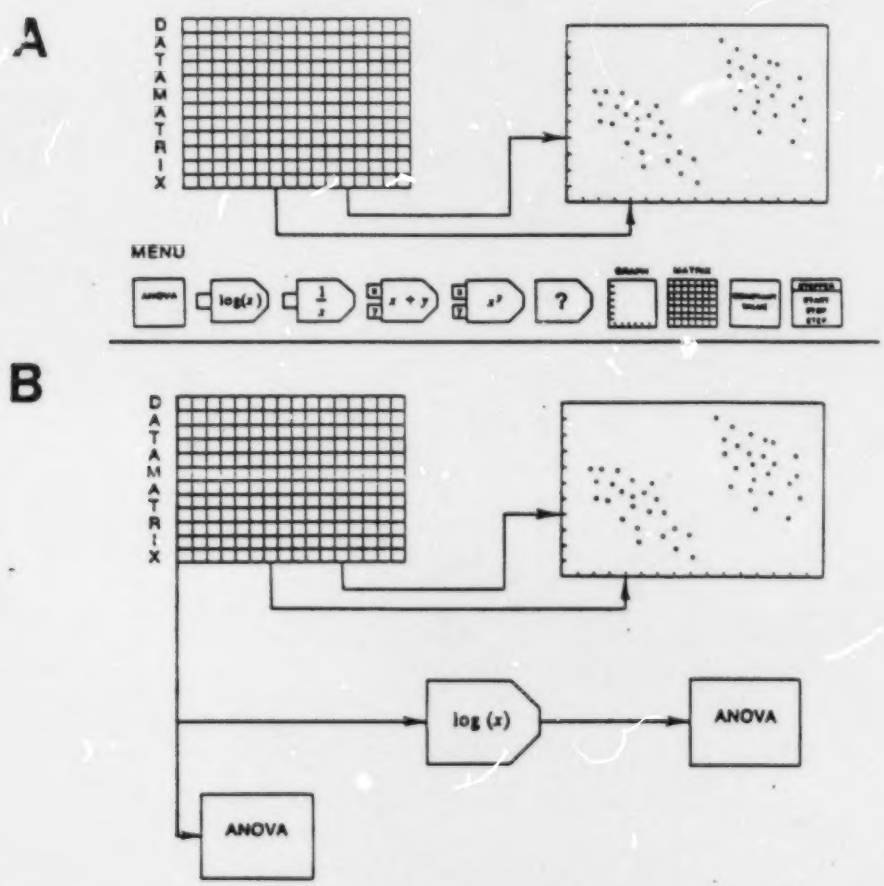

FIGURE 1
An elementary example of doing simple statistical computations by Direct Manipulation. A: The basic components: The data are contained in the matrix, represented by the icon in the upper left corner of the screen. At the bottom of the screen are basic icons that represent possible functions. To use one, a copy of the desired icon is moved to the screen and connected up, much as is shown for the graph. B: More complex interconnections, including the use of a logarithmic transformation of the data, a basic statistical package (for means and standard deviations), and an Analysis of Variance Package (ANOVA).

task domain, but only minimal knowledge of the computer or of computing.

> *A number of direct manipulation systems already exist. One was described briefly in Norman's chapter on 'Cognitive Engineering': Bill Budge's Pinball Construction Set (Budge, 1983). It allows for the construction of game environments simply by moving the desired pieces onto the game, then stretching, coloring, and otherwise manipulating the environment.*

Direct manipulation interfaces seem remarkably powerful. Shneiderman has suggested that they have the following virtues:

BEST COPY AVAILABLE

BEST COPY AVAILABLE

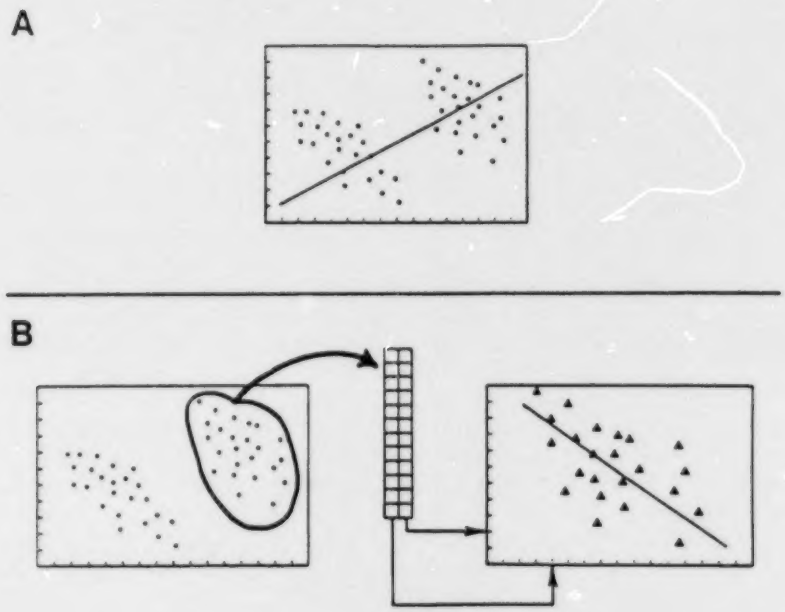

FIGURE 2
A: The scatter plot formed by plotting the two columns against against each other, along with the best fitting regression line to the data. It is clear that the data really fall into two quite distinct clusters and that it would be best to look at these clusters independently. B: The clusters are analyzed by circling the desired data, then treating each group of circled data as if it were a new matrix of values, which can be treated as a data source and analyzed in standard ways.

1. *Novices can learn basic functionality quickly, usually through a demonstration by a more experienced user.*
2. *Experts can work extremely rapidly to carry out a wide range of tasks, even defining new functions and features.*
3. *Knowledgeable intermittent users can retain operational concepts.*
4. *Error messages are rarely needed.*
5. *Users can see immediately if their actions are furthering their goals, and if not, they can simply change the direction of their activity.*
6. *Users have reduced anxiety because the system is comprehensible and because actions are so easily reversible.* (Shneiderman, 1982, p. 251)

Can this really be true? Certainly there must be problems as well as benefits. It turns out that the concept of Direct Manipulation is complex. Moreover, although there are important benefits, there are also deficits. There are limitations: Like everything else, direct manipulation systems trade off one set of virtues and vices against another. It is important that we understand all the tradeoffs.

A Brief History of Direct Manipulation

The term "Direct Manipulation" was invented by Shneiderman (see Shneiderman, 1974, 1982, 1983; the 1983 review provides the best treatment of the topic). Shneiderman suggested "an integrated portrait of direct manipulation":

1. *Continuous representation of the object of interest.*
2. *Physical actions or labeled button presses instead of complex syntax.*
3. *Rapid incremental reversible operations whose impact on the object of interest is immediately visible.* (Shneiderman, 1982, p. 251)

Hints of direct manipulation programming environments have been around for quite some time. The first major landmark is Sutherland's *Sketchpad*, a graphical design program (Sutherland, 1963). Sutherland's goal was to devise a program that would make it possible for a person and a computer "to converse rapidly through the medium of line drawings." Sutherland's work is a landmark not only because of historical priority but because of the ideas that he helped develop: He was one of the first to discuss the power of graphical interfaces, the conception of a display as "sheets of paper," the use of pointing devices, the virtues of constraint representations, and the importance of depicting abstractions graphically.

Sutherland's ideas took 20 years to have widespread impact. The lag is perhaps due more to hardware limitations than anything else. Highly interactive, graphical programming requires the ready availability of considerable computational power and it is only recently that machines capable of supporting this type of computational environment have become inexpensive enough to be generally available. But now we see these ideas in many of the computer-aided design and computer-aided manufacturing systems; many of which can trace their heritage directly to Sutherland's work. Borning's *ThingLab* program (1979) explored a general programming environment, building upon many of the Sutherland's ideas within the Smalltalk programming environment. More recently Direct Manipulation systems have been appearing with reasonable frequency. We have already discussed *Bill Budge's Pinball Construction Set* (Budge, 1983). Other examples exist in the area of intelligent training systems (e.g., STEAMER: Hollan, Stevens, & Williams, 1980; Hollan, Hutchins, & Weitzman, 1984). Steamer makes use of similar techniques and also provide tools for the construction of interactive graphical interfaces. Finally, the spreadsheet program, first available on the Apple II computer and now commonplace, incorporates many of the essential features of Direct Manipulation. In the lead article of *Scientific American*'s special issue on computer software, Kay (1984) claims that the development of dynamic spreadsheet systems gives strong hints that programming styles are in the offing that will make programming as it has been done for the past 40 years obsolete.

The Goal: A Cognitive Account of Direct Manipulation

We see promise in the notion of direct manipulation, but as yet we see no explanation of it. There are systems with attractive features, and claims for the benefits of systems that give the user a certain sort of feeling, and even lists of properties that seem to be shared by systems that provide that feeling, but no account of how particular properties might produce the feeling of directness. The purpose of this chapter is to examine the underlying basis for Direct Manipulation systems. On the one hand, what is it that provides the feeling of "directness"? Why do the examples feel so natural? What is so compelling about the notion? On the other hand, why does it seem so painful at times? Certainly you wouldn't want to do basic programming this way, or would you? What about Norman's complaint (Norman, in press):

> *Much as I enjoy manipulating the parts of the pinball sets, much as my 4 year old son could learn to work it with almost no training or bother, neither of us are any good at constructing pinball sets. I can't quite get the parts in the right places: When I stretch a part to change its shape, I usually end up with an unworkable part. Balls get stuck in weird corners. The action is either too fast or too slow. Yes, it is easy to change each problem as it is discovered, but the number seems endless. I wish the tools were more intelligent—do as I am intending, not as I am doing.*

For us, the notion of *Direct Manipulation* is not a unitary concept nor even something that can be quantified in itself. It is an orienting notion. "Directness" is an impression or a feeling

about an interface. What we seek to do here is to characterize the space of interfaces and see where within that picture the range of phenomena that contribute to the feeling of directness might reside. The goal is to give cognitive accounts of these phenomena. At the root of our approach is the assumption that the feeling of directness results from the commitment of fewer cognitive resources. Or put the other way round, the need to commit additional cognitive resources in the use of an interface leads to the feeling of indirectness. As we shall see, some of the production of the feeling of directness is due to adaptation by the user, so that the designer can neither completely control the process, nor take full credit for the feeling of directness that may be experienced by the user.

There is a need to provide a characterization of how interfaces work, both in the computer science and the psychological senses, that will allow us to examine their strengths, weaknesses, and limitations. The computer science sense of "how they do it" concerns the computer-based technologies that contribute to interface power: object-oriented programming, constraint-based systems, logic programming, pointing devices, window systems, etc. In this chapter we focus primarily on the psychological sense of "how they do it" because we feel that this is where the power lies. We believe that while many technological advances may be helpful (new types of input devices, for example), even the existing technology is not used to full advantage because of a lack of understanding of the psychological side of interface design.

We will not attempt to set down hard and fast criteria under which an interface can be classified as direct or not direct. The sensation of directness is always relative. It is often due to the interaction of a number of factors. There are costs associated with every factor that increases the sensation of directness. At present we know of no way to measure the tradeoff values, but we will attempt to provide a framework within which one can say what is being traded off against what.

TWO ASPECTS OF DIRECTNESS: DISTANCE AND ENGAGEMENT

There are two separate and distinct aspects of the feeling of directness. One involves a notion of the distance between one's thoughts and the physical requirements of the system under use. A short distance means that the translation is simple and straightforward, that thoughts are readily translated into the physical actions required by the system and that the system output is in a form readily interpreted in terms of the goals of interest to the user. We call this aspect "*distance*" to emphasize the fact that directness is never a property of the interface alone, but involves a relationship between the task the user has in mind and the way that task can be accomplished via the interface. Here the critical issues involve minimizing the effort required to bridge the gulf between the user's goals and the way they must be specified to the system.

The second aspect of directness concerns the qualitative feeling of engagement, the feeling that one is directly manipulating the objects of interest. There are two major metaphors for the nature of human-computer interaction, a conversation metaphor and a model world metaphor. In a system built on the conversation metaphor, the interface is a language medium in which the user and system have a conversation about an assumed, but not explicitly represented world. In this case, the interface is an implied intermediary between the user and the world about which things are said. In a system built on the model world metaphor, the interface is itself a world where the user can act, and that changes state in response to user actions. The world of interest i explicitly represented and there is no intermediary between user and world. Appropriate use of the model world metaphor can create the sensation in the user of acting upon the objects of the task domain themselves. We call this aspect "*direct engagement.*"

Distance

An interfaces introduces distance to the extent there are gulfs between a person's goals and knowledge and the level of description provided by the systems with which the person must deal. These are the *Gulf of Execution* and the *Gulf of Evaluation* (Figure 3) that were incorporated by Norman in his chapter on Cognitive Engineering (Norman, in press). The Gulf of

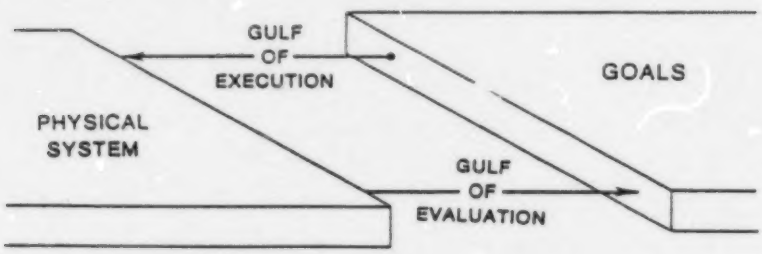

FIGURE 3
The Gulfs of Execution and Evaluation. Each gulf is unidirectional: the Gulf of Execution goes from goals to system state; the Gulf of Evaluation goes from system state to goals. (From Norman, in press.)

Execution is bridged by making the commands and mechanisms of the system match the thoughts and goals of the user as much as possible. The Gulf of Evaluation is bridged by making the output displays present a good Conceptual Model of the system that is readily perceived, interpreted, and evaluated. The goal in both cases is to minimize cognitive effort.

We suggest that the feeling of directness is inversely proportional to the amount of cognitive effort it takes to manipulate and evaluate a system and, moreover, that cognitive effort is a direct result of the Gulfs of Execution and Evaluation. The better the interfaces to the system help bridge the gulfs, the less cognitive effort needed, the more direct the resulting feeling of interaction.

Norman (in press) suggests that it is useful to consider an approximate theory of action, and in particular, that it is useful to consider separately the seven different stages of activity shown in Figure 4 that a person might go through in the performance of a task. For the current discussion, the important point is that the interface must try to match the goals of the person.

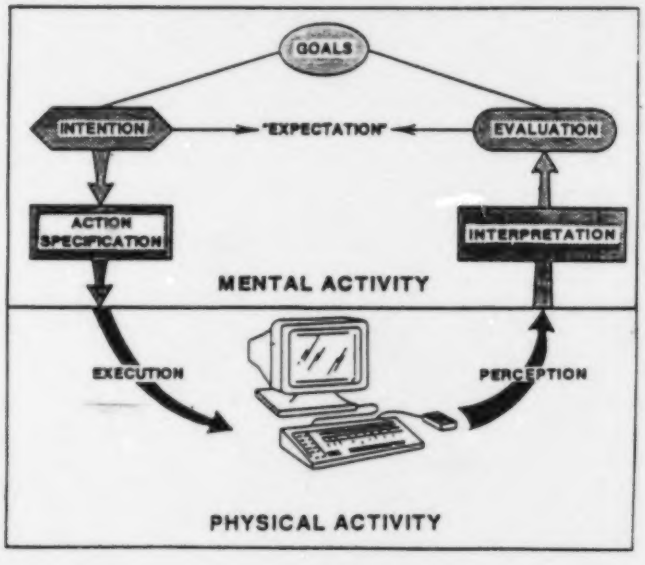

FIGURE 4
The seven stages of user activities involved in the performance of a task. (From Norman, in press.)

BEST COPY AVAILABLE

The two gulfs are bridged in two directions; from the system side by the system interface and from the user side by mental structures and interpretation. The bridges from the user's side are aided through the development of appropriate conceptual models—the design model and the user model discussed in Norman (in press), which are in turn aided by the system image presented by the interface. But a major burden should be carried by the interface itself. The more of the gulf spanned by the interface, the less distance need be bridged by the efforts of the user.

Direct Engagement

The description of the nature of interaction to this point begins to suggest how to make a system less difficult to use, but it misses an important point, a point that is the essence of Direct Manipulation. The analysis of the execution and evaluation process explains why there is difficulty in using a system and it says something about what must be done to minimize the mental effort required to use a system. But there is more to it than that. The systems that best exemplify Direct Manipulation all give us the qualitative feeling that we are directly engaged with control of the objects—not with the programs, not with the computer, but with the semantic objects of our goals and intentions. This is the feeling that Laurel (in press) discusses: a feeling of first-personness, of direct engagement with the objects that concern us. Are we analyzing data? Then we should be manipulating the data themselves or if we are designing an analysis of data, we should be manipulating the analytic structures themselves. Are we playing a game? Then we should be manipulating directly the game world, touching and controlling the objects in that world, with the output of the system responding directly to our actions, and in a form compatible with them.

Historically, most interfaces have been built on the conversational metaphor. There is power in the abstractions that language provides (we discuss some of this later), but the implicit role of interface as an intermediary to a hidden world denies the user direct engagement with the objects of interest. Instead, the user is in direct contact with linguistic structures, structures that can be interpreted as referring to the objects of interest, but that are not those objects themselves. We argue that the central metaphor of the interface should be that of the model world: Instead of describing the actions of interest, the user should be doing them. In the conventional interface, the system describes the results of the actions: In the model world the system would present directly the actions taken upon the objects. This change in central metaphor is made possible by relatively recent advances in technology. One of the exciting prospects for the study of Direct Manipulation is the exploration of the properties of systems that provide for direct engagement.

Building interfaces on the model world metaphor requires a special sort of relationship between the input interface language and the output interface language. In particular, the output language must represent its subject of discourse in a way that natural language does not normally do. The expressions of a Direct Manipulation output language must behave in such a way that the user can assume that they, in some sense, *are* the things they refer to. Furthermore, the nature of the relationship between input and output language must be such that an output expression can serve as a component of an input expression. When these conditions are met, it is as if we are directly manipulating the things that the system represents.

Thus, consider a system in which a file is represented by an image on the screen and actions are done by pointing to and manipulating the image. In this case, if we can specify a file by pointing at the screen representation, we have met the goal that the output "language" (in this case an image) be allowed as a statement in the input expression (in this case, by pointing at the screen representation). If we ask for a listing of files, we would want the result to be a physical display that can, in turn, be used directly to specify the further operations to be done. Notice that this is not how a conversation works. In conversation, one may refer to what has previously been said, but one cannot operate upon what has been said. This kind of requirement does not require an interface of pictures, diagrams, or icons. It can be done with words and descriptions. The key properties are that the objects, whatever their form, have behaviors,

can be referred to by other objects, and that referring to an object causes it to behave. In the file listing example, we must be able to use the output expression that represents the file in question as a part of the input expression calling for whatever operation we desire upon that file, and the output expression that represents the file must change as a result of being referred to in this way. The goal is to permit the user to act as if the representation is the thing itself.

These conditions are met in many screen editors when the task is the arrangement of strings of characters. The characters appear as they are typed. They are then available for further operations. We treat them (but not their meanings!) as though they are the things we are manipulating. These conditions are also met in the statistics example with which we opened this chapter (Figure 1), in Steamer, and by the *Pin-Ball Construction Set*. The special conditions are not met in file listing commands on most systems, the commands that allow one to display the names and attributes of file structure. The issue is that the outputs of these commands are "names" of the objects. Operating on the names does nothing to the objects to which they refer. In a Direct Manipulation situation, we would feel that we had the files in front of us, that the program that "listed" the files actually placed the files before us: Any further operation on the files would take place upon the very objects delivered by the directory listing command. This would provide the feeling of directly manipulating the objects that were returned.

> *When we deal with the conceptually complex objects of computation, we have to remember that each object typically has a number of different attributes. An object can be viewed from different "perspectives," behaving quite differently with each viewpoint.*
>
> *Consider the file example. If the objects are the names of the files (or icons containing the names), this depiction will work well as long as the operations to be performed are things like removing the files, renaming them, or positioning them as sources of input or receivers of output. But it is not a good representation for all possible operations. Changing the contents of the file, for instance, requires a representation of the contents as opposed to its "surface" features: The objects to be manipulated are the words or images within the file itself. This requires a different representation. Or consider what is required to change the properties of the file—its protection status, perhaps. Here we require a representation that makes visible and manipulable these properties of the file. Thus, at least three different representations are required. It is unlikely and probably inappropriate to try to develop a single representation of an object that can serve all purposes.*
>
> *We should recognize that more than one view of an object is often necessary and provide a separate representation for each one. Even a system that gives us the illusion that we are manipulating the objects directly needs different representations when the objects are viewed from different perspectives.*

The point is that when an interface presents a world of action rather than a language of description, manipulating a representation can have the same effects and the same feel as manipulating the thing being represented. The members of the audience of a well staged play willfully suspend their beliefs that the players are actors and become directly engaged in the content of the drama. In a similar way, the user of a well designed *model world* interface can willfully suspend belief that the objects depicted are artifacts of some program and can thereby directly engage the world of the objects. This is the essence of the "first-personness" feeling of direct engagement. Let us now return to the issue of distance and explore the ways that an interface can be direct or indirect with respect to a particular task.

TWO FORMS OF DISTANCE: SEMANTIC AND ARTICULATORY

Whenever we interact with a device, we are using an interface language. That is, we must use a language to describe to the device the nature of the actions we wish to have performed. This is true regardless of whether we are dealing with an interface based on the conversation metaphor or on the model world metaphor, although the properties of the language in the two cases are different. Such a description is an expression in the interface language.

The notion of an interface language is not confined to the everyday meaning of language. Setting a switch or turning a steering wheel can be expressions in an interface languages if switch-setting or wheel-turning are how one specifies the operations that are to be done. After an action has been performed, evaluation of the outcome requires that the device make available some indication of what has happened: That output is an expression in the output interface language. Output interface languages are often impoverished. Frequently the output interface language does not share vocabulary with the input interface language. Two forms of interface language—two dialects, if you will—must exist to span the gulfs between user and device: The input interface language and the output interface language.

Both the languages people speak and computer programming languages are almost entirely symbolic in the sense that there is an arbitrary relationship between the form of a vocabulary item and its meaning. The reference relationship is established by convention and must be learned. There is no way to infer meaning from form for most vocabulary items. Because of the relative independence of meaning and form we describe separately two properties of the interface language: *Semantic Directness* and *Articulatory Directness*.

Figure 5 summarizes the relationship between semantic and articulatory distance. We now examine semantic and articulatory directness separately for the Gulfs of Execution and Evaluation.

Semantic Directness

Semantic directness concerns the relation of the meaning of an expression in the interface language to what the user wants to say. Two important questions about semantic directness are:

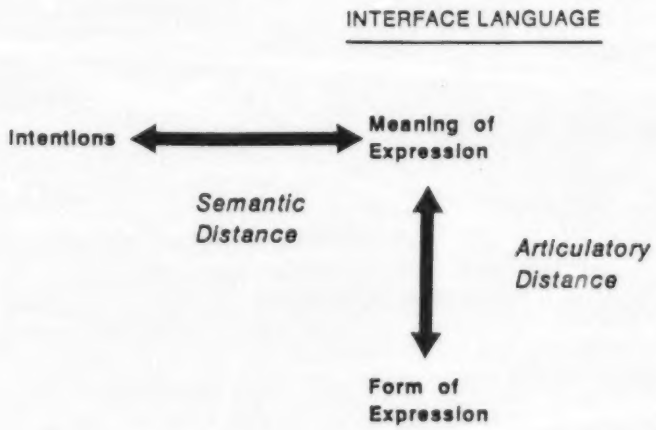

FIGURE 5
Every expression in the interface language has a meaning and a form. Semantic distance reflects the relationship between the user intentions and the meaning of expressions in the interface languages both for input and output. Articulatory distance reflects the relationship between the physical form of the interaction language and its meaning, again, both for input and output. The easier it is to go from the form or appearance of the input or output to meaning, the smaller the articulatory distance.

BEST COPY AVAILABLE

- *Is it possible to say what one wants to say in this language?* That is, does the language support the user's conception of the task domain? Does it encode the concepts and distinctions in the domain in the same way that the user thinks about them?

- *Can the things of interest be said concisely?* Can the user say what is wanted in a straightforward fashion, or must the user construct a complicated expression to do what appears in the user's thoughts as a conceptually simple piece of work?

Semantic directness is an issue with all languages. Natural languages generally evolve such that they have rich vocabularies for domains that are of importance to their speakers. When a person learns a new language—especially when the language is from a novel culture—the new language may seem indirect, requiring complicated constructs to describe things the learner thinks should be easy to say. But the differences in apparent directness reflect differences in what things are thought important in the two cultures. Natural languages can and do change as the need arises. This occurs through the introduction of new vocabulary or by changing the meaning of existing terms. The result is to make the language semantically more direct with respect to the topic of interest.

Semantic Distance in the Gulfs of Execution and Evaluation

> *Beware the Turing tar-pit in which everything is possible but nothing of interest is easy* (Alan Perlis, 1982).

The Gulf of Execution. At the highest level of description, a task may be described by the user's intention: "compose this piece" or "design this building." At the lowest level of description, the performance of the task consists of the shuffling of bits around inside the machine. Between the interface and the low-level operations of the machine is the system-provided task-support structure that implements the expressions in the interface language. The situation that Perlis called the "Turing tar-pit" is one in which the interface language lies near or at the level of bit shuffling of a very simple abstract machine. In this case, the entire burden of spanning the gulf from user intention to bit manipulation is carried by the user. The relationship between the user's intention and the organization of the instructions given the machine is distant, complicated, and hard to follow. Where the machine is of minimal complexity, as is the case with the Turing machine example, the wide gulf between user intention and machine instructions must be filled by the user's extensive planning and translation activities. These activities are difficult and rife with opportunities for error.

Semantic directness requires matching the level of description required by the interface language to the level at which the person thinks of the task. It is always the case that the user must generate some information-processing structure to span the gulf. Semantic distance in the gulf of execution reflects how much of the required structure is provided by the system and how much by the user. The more that the user must provide, the greater the distance to be bridged.

The Gulf of Evaluation. On the evaluation side, semantic distance refers to the amount of processing structure that is required for the user to determine whether the goal has been achieved. If the terms of the output are not those of the user's intention, the user will be required to translate the output into terms that are compatible with the intention in order to make the evaluation. For example, suppose a user's intent is to control how fast the water level in a tank rises. The user does some controlling action and observes the output. But if the output only shows the current value, the user has to observe the value over time and mentally compare the values at different times to see what the rate of change is: See Figure 6A. The information needed for the evaluation is in the output, but it is not there in a form that directly fits the terms of the evaluation. The burden is on the user to perform the required transformations and that requires effort. Suppose the rate of change were directly displayed, as in Figure 6B. This indication reduces the mental workload, making the semantic distance

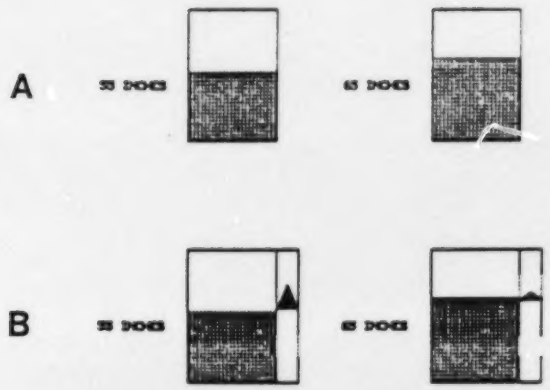

FIGURE 6
Matching user's intentions by appropriate output language. The user attempts to control the rate at which the water level in the tank is rising. In A, the only indication is a meter that shows the current level. This requires the user to observe the meter over time and to do a mental computation on the observations. B shows a display that is more semantically direct: The rate of change is directly indicated, both numerically and graphically. (These illustrations are from the working STEAMER system of Hollan, Hutchins, and Weitzman, 1984).

between intentions and output language much shorter.

Reducing the Semantic Distance That Must Be Spanned

Figure 6 provides one illustration of how semantic distance can be changed. In general, there are only two basic ways to reduce the distance, one from the system side (requiring effort on the part of the system designer), the other from the user side (requiring effort on the part of the user). Each direction of bridge building has several components. Here let us consider the following possibilities:

- The designer can construct higher-order and specialized languages that move toward the user, making the semantics of the input and output languages match that of the user;
- The user can develop competence by building new mental structures to bridge the gulfs. In particular, this requires the user to automate the response sequence and to learn to think in the same language as that required by the system.

Higher-level languages. One way to bridge the gulf between the intentions of the user and the specifications required by the computer is well-known: Provide the user with a higher-order language, one that directly expresses frequently encountered structures of problem decomposition. Instead of requiring the complete decomposition of the task all the way to low-level operations, let the task be described in the same language used within the task domain itself. Although the computer still requires low-level specification, the job of translating from the domain language to the programming language can be taken over by the machine itself.

This implies that designers of higher-level languages should consider how to develop interface languages for which it will be easy for the user to create the mediating structure between intentions and expressions in the language. One way to facilitate this process is to provide consistency across the interface surface. That is, if the user builds a structure to make contact with some part of the interface surface, a savings in effort can be realized if it is possible to use all or part of that same structure to make contact with other areas.

Beware the over-specialized system where operations are easy, but little of interest is possible (the converse of the Turing tar-pit).

The result of matching a language to the task domain brings both good news and bad news. The good news is that tasks are easier to specify. Even if considerable planning is still required to express a task in a high-level language (Riley & O'Malley 1984; Riley, in press), the amount of planning and translation that can be avoided by the user and passed off to the machine is enormous. That's the good news. The bad news is that the language has lost generality. Tasks that do not easily decompose into the terms of the language may be difficult or impossible to represent: In the extreme case, what can be done is easy to do, but outside that specialized domain, nothing can be done. The power of a specialized language system derives from carefully specified primitive operations, selected to match the predicted needs of the user, thus capturing frequently occurring structures of problem decomposition.

The trouble is that there is a conflict between generality and matching to any specific problem domain. Some high-level languages and operating systems have striven to close the gap between user intention and the interaction language while preserving freedom and ease of general expression by allowing for extensibility of the language or operating system. Such systems allow the users to move the interface closer to their conception of the task.

> *The Lisp language and the UNIX operating system serve as examples of this phenomenon. Lisp is a general purpose language, but one that has extended itself to match a number of special, high-level domains. As a result, Lisp can be thought of as a system rather than a language, with numerous levels layered on top of the underlying language kernel. There is a cost to this method. As more and more specialized domain levels get added, the language system gets larger and larger, becoming more clumsy to use, more expensive to support, and more difficult to learn. Just look at any of the manuals for the large Lisp systems (Interlisp, Zetalisp) to get a feel for the complexity involved. The same is true for UNIX operating system which started out with a number of low-level, general primitive operations. Users were allowed (and encouraged) to add their own, more specialized operations, or to package together the primitives into higher-level operations. The results in all these cases are massive systems that are hard to learn and that require a large amount of support facilities. The documentation becomes huge, and not even system experts know all that is present. Moreover, the difficulty of maintaining such a large system increases the burden on everyone, and the possibility of having standard interfaces to each specialized function has long been given up.*

The point is that as one approaches the user's intention end of the gulf, functions become more complicated and more specialized in purpose. Because of the incredible variety of human intentions, the lexicon of a language that aspires to both generality of coverage and domain specific functions can grow very large. In any of the modern dialects of Lisp one sees a microcosm of the argument about high-level languages in general. The fundamentals of the language are simple, but a great deal of effort is required to do anything useful at the low level of the language itself. Higher-level functions written in terms of lower-level ones make the system easier to use when the functions match intentions, but in doing so they may restrict possibilities, proliferate vocabulary, and require that a user know an increasing amount about the language of interaction rather than the domain of action.

Make the output show semantic concepts directly. An example of reducing semantic distance on the output side is provided by the scenario of controlling the rate of filling a water tank, described in Figure 6. In that situation, the output display was modified to show rate of flow directly, something normally not displayed but instead left to the user to compute mentally.

In similar fashion, the change from line-oriented text editors to screen-oriented text editors, where the effects of editing commands could be seen instantly, is another example of matching

the display to the user's semantics. In general, the development of WYSIWYG systems ("What you see is what you get") provides other examples. And finally, the spreadsheet program has been so important, in part because its output format continually and elegantly shows the state of the system in the very terms the accountant likes to use, at least so some believe. This format is actually not so good for many users, which explains the rapid proliferation of graphical packages that transform spreadsheet data into a form more readily interpretable by a wider class of user.

The attempt to develop good semantic matches at the system output confronts the same conflict between generality and power faced in the design of input languages. If the system is too specific and specialized, the output displays lack generality. If the system is too rich, the user has trouble learning and selecting among the possibilities. One solution for both the output and input problem is to abandon hope of maintaining general computing and output ability and to develop special purpose systems for particular domains or tasks. In such a world, the location of the interface in semantic space is pushed closer to the domain language description. Here, things of interest are made simple because the lexicon of the interface language maps well into the lexicon of domain description. Considerable planning may still go on in the conception of the domain itself, but little or no planning or translation is required to get from the language of domain description to the language of the interface. The price paid for these advantages is a loss of generality: Many things are unnatural or even impossible.

Automated behavior does not reduce semantic distance. Cognitive effort is required to plan a sequence of actions to satisfy some intent. Generally, the more structure required of the user, the more effort use of the system will entail. However, this gap can be overcome if the users become familiar enough with the system. Structures that are used frequently need not be rebuilt every time they are needed if they have been remembered. Thus, a user may remember how to do something rather than having to rederive how to do it. It is well known that when tasks are practiced sufficiently often, they become automated, requiring little or no conscious attention. As a result, over time, the use of the interface to solve a particular set of problems will feel less difficult and more direct. Experienced users will sometimes argue that the interface they use directly satisfies their intentions, even when less skilled users complain of the complexity of the structures. To the skilled user, the interface feels direct because the invocation of mediating structure has been automated. They have learned how to transform frequently arising intentions into action specifications. The result is a feeling of directness as compelling as that which results from semantic directness. As far as the user is concerned, the intention comes to mind and the action gets executed. There are no conscious intervening stages.

> *The point was made forcibly to us in a class in which these ideas about Direct Manipulation were being described. Erasing a word by using an eraser was defined as direct action. Erasing by appropriate marking with a mouse pointing device in a text editor was also called direct, although slightly less. But using a standard, command-language based text editor was not direct. "But why not?" queried a student. "I am an expert user of vi, and when I wish to delete a word, all I do is think 'delete that word,' my fingers automatically type 'dw,' and the word disappears from the screen. How could anything be more direct?"*

The frequent use of even a poorly designed interface can sometimes result in a feeling of directness like that produced by a semantically direct interface. A user can compensate for the deficiencies of the interface through continual use and practice so that the ability to use it becomes automated, requiring little conscious activity. While automation is one factor which can contribute to a feeling of directness, it is essential for an interface designer to distinguish it from semantic directness. Automatization does not reduce the semantic distance that must be spanned. The gulfs between a user's intentions and the interface must still be bridged by the user. Although practice and the resulting expertise can make the crossing less difficult, it does not reduce the magnitude of the gulfs. Planning activity may be replaced by a single memory

retrieval so that instead of figuring out what to do, the user remembers what to do. Automatization may feel like direct control, but it comes about for completely different reasons than semantic directness. Automatization is useful, for it improves the interaction of the user with the system, but the feeling of directness it produces depends only upon how much practice the particular user has: it gives the system credit for the work the user has done. Although we need to remember that this happens, that users may adjust themselves to the interface and, with sufficient practice, may view it as directly supporting their intentions, we need to distinguish between the cases in which the feeling of directness originates from a close semantic coupling between intentions and the interface language and that which originates from practice. The resultant feeling might be the same in the two cases, but there are crucial differences between how the feeling gets acquired and what one needs to do as an interface designer to generate it.

The user can adapt to the system representation. One way to span the gulf is for the users to change their own conceptualization of the problem so that they come to think of it in the same terms as the system. In some sense, this means that the gulf is bridged by moving the user closer to the system. Because of their experience with the system, the users change both their understanding of the task and the language with which they think of the issues. This is related to the notion of linguistic determinism. If it is true that the way we think about something is shaped by the vocabulary we have for talking about it, then it is important for the designer of a system to provide the user with a good representation of the task domain in question. The interface language should provide a powerful, productive way of thinking about the domain.

> *This is the Whorfian hypothesis of language, transformed to the computer setting. The basic idea is that dedicated users change the very language and manner with which they think about problems, so that the initial goals and intentions are formed in the same language required by the system, thus bridging the gulf from the user side. One has only to listen to devoted system users to be convinced of the validity of the concept: Indeed, the actions available on the computer system have been so thoroughly introduced into the culture of the user, that the concepts are used in non-computer applications as well, where they are no longer directly applicable:*
>
> *"What movies are playing in town?* Grep *the newspaper, would you?"*
>
> *"Both those ideas are good: Let's* cons *them up."*

Linguistic determinism takes place at a more fundamental level than the other ways of reducing the semantic distance. While moving the interface closer to the user's intentions may make it difficult to realize some intentions, changing the user's conception of the domain may prevent some intentions from arising at all. So while a well designed special purpose language may give the user a powerful way of thinking about the domain, it may also restrict the user's flexibility to think about the domain in different ways.

That the user may change conceptual structure to match the interface language follows from the notion that every interface language implies a representation of the tasks it is applied to. The representation implied by an interface is not always a coherent one. Some interfaces provide a collection of partially overlapping views of a task domain. If the user is to move toward the model implicit in the interface, and thus reduce the semantic distance, that model should be coherent and consistent over some conception of the domain. There is, of course, a tradeoff here between the costs to the user of learning a new way to think about a domain and the potential added power of thinking about it in the new way.

Virtuosity and semantic distance. Sometimes users have a conception of a task and of a system that is broader and more powerful than that provided by an interface. The structures they build to make contact with the interface go beyond it. This is how we characterize virtuoso performances in which the user may "misuse" limited interface tools to satisfy intentions that even the system designer never anticipated. In such cases of virtuosity the notion of semantic distance becomes more complicated and we need to look very carefully at the task

that is being accomplished. Semantic directness always involves the relationship between the task one wishes to accomplish and the ways the interfaces provides for accomplishing it. If the task changes, then the semantic directness of the interface may also change.

Consider a musical example: Take the task of producing a middle-C note on two musical instruments, a piano and a violin. For this simple task, the piano provides the more direct interface because all one need do is find the key for middle-C and depress it, whereas on the violin, one must place the bow on the G string, place a choice of fingers in precisely the right location on that string, and draw the bow. A piano's keyboard is more semantically direct than the violin's strings and bow for the task of producing notes. The piano has a single well-defined vocabulary item for each of the notes within its range, while the violin has an infinity of vocabulary items many of which do not produce proper notes at all. However, when the task is playing a musical piece well rather than simply producing notes, the directness of the interfaces can change. In this case, one can complain that a piano has a very indirect interface because it is a machine with which the performer "throws hammers at strings." The performer has no direct contact with the components that actually produce the sound and so, the production of desired nuances in sound is more difficult. Here, as musical virtuosity develops, the task that is to be accomplished also changes from just the production of notes to concern for how to control more subtle characteristics of the sounds like vibrato, the slight changes in pitch used to add expressiveness. For this task the violin provides a semantically more direct interface than the piano. Thus, as we have argued earlier, an analysis of the nature of the task being performed is essential in determining the semantic directness of an interface.

> *We are reminded of a quote from Minsky (1967):*
>
> *"A computer is like a violin. You can imagine a novice trying first a phonograph and then a violin. The latter, he says, sounds terrible. That is the argument heard from our humanists and most of our computer scientists. Computer programs are good they say, for particular purposes, but they aren't flexible. Neither is a violin, or a typewriter, until you learn how to use it."*

Articulatory Directness

In addition to its meaning, every vocabulary item in every language has a physical form and that form has an internal structure. Words in natural languages, for example, have phonetic structure when spoken and typographic structure when printed. Similarly, the vocabulary items that constitute an interface language have a physical structure. Where *semantic directness* has to do with the meanings of the expressions and their semantic complexity, *articulatory directness* has to do with the physical *form* of the vocabulary items. On the input side, the form may be a sequence of character-selecting key presses for a command language interface, the movement of a mouse and the associated "mouse clicks" in a pointing device interface, or a phonetic string in a speech interface. On the output side, the form might be a string of characters, a change in an iconic shape, an auditory signal, or a graph, diagram, or animation.

There are ways to design languages such that the relationships between the forms of the vocabulary items and their meanings are not arbitrary. One technique is to make the physical form of the vocabulary items structurally similar to their meanings. In spoken language this relationship is called onomatopoeia. Onomatopoetic words in spoken language refer to their meanings by imitating the sound they refer to. Thus we talk about the "boom" of explosions or the "cock-a-doodle-doo" of roosters. There is an economy here in that the user's knowledge of the structure of the surface acoustical form has a non-arbitrary relation to meaning. There is a directness of reference in this imitation. An intervening level of arbitrary symbolic relations is eliminated. Other uses of language exploit this effect partially. Thus, while the word "long" is arbitrarily associated with its meaning, sentences like "She stayed a looooooooooong time" exploit a structural similarity between the surface form of "long" (whether written or spoken) and the intended meaning. The same sorts of things can be done in the design of

interface languages.

> *In many ways, the interface languages should have an easier time of exploiting articulatory similarity than do natural languages because of the rich technological base available to them. Thus, if the intent is to draw a diagram, the interface might accept as input the drawing motions. In turn, it could present as output diagrams, graphs, and images. If one is talking about sound patterns to the input interface language, the output could be the sounds themselves. The computer has the potential to exploit articulatory similarities through technological innovation in the varieties of dimensions upon which it can operate. This potential has not been exploited, in part because of economic constraints. The restriction to simple keyboard input limits the form and structure of the input languages and the restriction to simple, alphanumeric terminals with small, low resolution screens, limits the form and structure of the output languages.*

Articulatory Distance in the Gulfs of Execution and Evaluation

The relationships among semantic distance, articulatory distance, and the gulfs of execution and evaluation are shown in Figure 7.

Take the simple, commonplace activity of moving a cursor on the screen. If we do this by

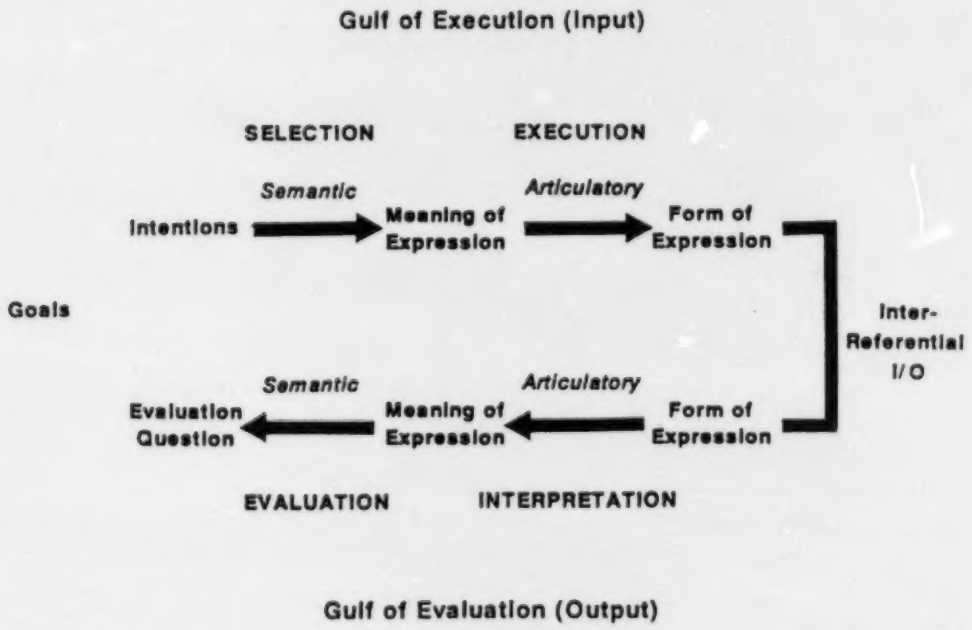

FIGURE 7
Bridging either gulf involves spanning both semantic and articulatory distance. In bridging the Gulf of Execution, the user *selects* an input expression with the desired meaning (spanning semantic distance), determines the form of the expression (spanning articulatory distance), and then *executes* the form of the expression. In bridging the Gulf of Evaluation, the form of the output expression is *interpreted* to yield its meaning (spanning articulatory distance), and the meaning is then compared to the intention in order to *evaluate* the outcome (spanning semantic distance). Where an input expression can incorporate or refer to an output expression, there is an inter-referential input/output relationship which may give the user a sense of directly manipulating the objects represented.

moving a mouse, pointing with the finger or a light pen at the screen, or otherwise mimicking the desired motion, then at the level of action execution, these are all articulatory direct interactions. The meaning of the intention is cursor movement and the action is specified by means of a similar movement. One way to achieve articulatory directness at the input side is to provide an interface that permits specification of an action by mimicking it, thus supporting a articulatory similarity between the vocabulary item and its meaning. Any non-arbitrary relationship between the form of an item and its meaning can be a basis for articulatory directness. Where it is not possible to create a non-arbitrary relationship between the form of the item and its meaning, we recognize that not all arbitrary relationships are equally easy to learn. It may be possible to exploit previous user knowledge in creating this relationship. The work on command names in command language interfaces is an instance of trying to develop memorable and discriminable arbitrary relationships between the forms and the meanings of command names.

Articulatory directness at the output side is similar. If the user is following the changes in some variable, a moving graphical display can provide articulatory directness. A table of numbers, although containing the same semantic information, is not articulatory direct. Thus, the graphical display and the table of numbers might both be equal in semantic directness, but unequal in articulatory directness. Articulatory direct interaction can couple the interaction between action and meaning so naturally that the relationships between intentions and actions and between actions and output seem straightforward and obvious.

> *Our students have pointed out that although it is indeed mimetically and articulatory direct to cause cursor movement by an analogous movement of a pointing device, in many instances the truly direct interface will do away with cursors. That is, cursor movements reflect current technological limitations.*
>
> *If we expand our imaginations, we can develop interface technologies that are even more direct than pointing movements. Thus, to delete a word, why move a cursor to the spot: Simply look at the word and speak the appropriate command.*

In general, articulatory directness is highly dependent upon I/O technology. To make actions and displays articulatory direct requires a much richer set of technological input/output devices than most systems currently have. We will, of course, need to have keyboards, pointing devices, and high-resolution, bit-mapped screens (which not all machines or terminals have today). The mouse is a *spatio-mimetic* device. That means that it can provide articulatory direct input for tasks that can be represented spatially. The mouse is useful for a wide variety of tasks not because of any properties inherent in itself, but because we map so many kinds of relationships (even ones that are not intrinsically spatial) on to a spatial metaphor. In addition, we need sound and speech, certainly as outputs, and possibly as inputs. Precise control of timing will be necessary for those applications where the domain of interest is time-sensitive. In general, we will want to open our imaginations to the set of relevant technologies. Perhaps it is stretching the imagination beyond its willing limits, but Francis Galton suggested and carried out a set of experiments on doing arithmetic by sense of smell (Galton, 1894). Less fancifully conceived, input might be sensitive not only to touch, place, and timing, but also to pressure or to torque. (See Buxton, in press.)

> *Articulatory distance can also be manipulated by the mental model adopted by the user. An example of this comes from the following story, told us by Yutaka Sayeki. His motorcycle had a switch on the left handlebar for controlling the turn-signals: Moving the switch forward signaled a right turn, backward a left turn. The switch control was semantically direct in that it had a single unambiguous item for each relevant intention regarding the turn signals, but Sayeki's understandings of the forms of the actions encouraged him to map the action "push switch forward" onto the intention "turn left," which is wrong. As a result, it was difficult to remember*

> *which switch direction was associated with which direction of turn. A mimetically direct switch would have moved to the right for the right turn, to the left for the left turn. Sayeki solved the problem through the creation of an appropriate mental model, in part by realizing that the motorcycle switch motion was analogous to the required movement of the turn signal lever in an automobile: The required direction is parallel to the direction of movement of the handlebar (or steering wheel).*
>
> *Consider the way in which the handlebars of the motorcycle turn. When making a left turn, the left handlebar moves backwards. For a right turn, the left handlebar moves forwards. The turn switch was located on the left handlebar, and the required switch movements exactly paralleled the handlebar movements. By reconceptualizing the task as signaling the direction of motion of the handlebars, the switch suddenly became mimetically, articulatory direct: The switch movement mimics the desired movement. With one mental model, the motion of the switch seems arbitrary, indirect, and difficult to remember. With a different mental model, the switch motion is articulatory direct and, therefore, easy to learn and to use.*

Iconographic languages. Pictographic and iconic languages are examples of articulatory representation in which the form of the expression is related to its meaning. By definition, an icon is a representation that stands for its object by virtue of a resemblance to it. However, even when the form of the icon is very like its intended meaning, the mapping is not complete. Instead, certain features of the referent are abstracted and preserved in the form of the icon, while others are discarded. And even those features that are preserved may be established by convention. For example, while the international iconic symbols for male and female (see pictures) may seem imitative of a fundamental distinction, they are scarcely interpretable to societies in which trousers and skirts are not worn. Even the interpretation of widely recognized icons requires background knowledge of conventions. Most icons of necessity are abstractions of the thing they depict. Thus, even iconic or pictographic representations are not totally articulatory direct: Their interpretation requires knowledge or explanation.

DIRECT ENGAGEMENT

Direct Engagement occurs when a user experiences direct interaction with the objects in a domain. Here there is a feeling of involvement directly with a world of objects rather than of communicating with an intermediary. The interactions are much like interacting with objects in the physical world. Actions apply to the objects, observations are made directly upon those objects, and the interface and the computer become invisible. Although we believe this feeling

of direct engagement to be of critical importance, in fact, we know little about the actual requirements for producing it. Laurel (in press), discusses some of the requirements. At a minimum, to produce a feeling of direct engagement the system needs:

- Execution and evaluation to be direct in the senses discussed in this chapter;
- Input and output languages of the interface to be inter-referential, allowing an input expression to incorporate or make use of a previous output expression. This is crucial for creating the illusion that one is directly manipulating the objects of concern;
- The system to be responsive, with no delays between execution and the results, except where those delays are appropriate for the knowledge domain itself;
- The interface to be unobtrusive, not interfering or intruding. If the interface itself is noticed, then it stands in a third-person relationship to the objects of interest, and detracts from the directness of the engagement.

In order to have a feeling of direct engagement, the interface must provide the user with a world in which to interact. The objects of that world must feel like they are the objects of interest, that one is doing things with them and watching how they react. In order for this to be the case, the output language must present representations of objects in forms that behave in the way that the user thinks of the objects behaving. Whatever changes are caused in the objects by the set of operations must be depicted in the representation of the objects. This use of the same object as both an input and output entity is essential to providing objects that behave as if they are the real thing. It is because an input expression can contain a previous output expression that the user feels the output expression is the thing itself and that the operation is applied directly to the thing itself. This is exactly the concept of "inter-referential I/O" discussed by Draper (in press).

In addition, all of the discussion of semantic and articulatory directness apply here too, because the designer of the interface must be concerned with what is to be done and how one articulates that in the languages of interaction. But the designer must also be concerned with creating and supporting an illusion. The specification of what needs to be done and evidence that it has been done must not violate the illusion, else the feeling of direct engagement will be lost.

One factor that seems especially relevant to maintaining this illusion is the form and speed of feedback. Rapid feedback in terms of changes in the behavior of objects not only allows for the modification of actions even as they are being executed, but also supports the feeling of acting directly on the objects themselves. It removes the perception of the computer as an intermediary by providing continual representation of system state. In addition, rapidity of feedback and continuous representation of state allows one to make use of perceptual faculties in evaluating the the outcome of actions. We can watch the actions take place, monitoring them much like we monitor our interactions with the physical world. The reduction in the cognitive load of mentally maintaining relevant information and the form of the interaction contribute to the feeling of engagement.

A SPACE OF INTERFACES

Distance and engagement are depicted in Figure 8 as two major dimensions in a space of interface designs. The dimension of engagement has two landmark values: One is the metaphor of interface as conversation; the other the metaphor of interface as model world. The dimension of distance actually contains two distances to be spanned: semantic and articulatory distances, the two kinds of gulfs that lie between the user's conception of the task and the interface language.

The least direct interface is often one that provides a low-level language interface, for this is

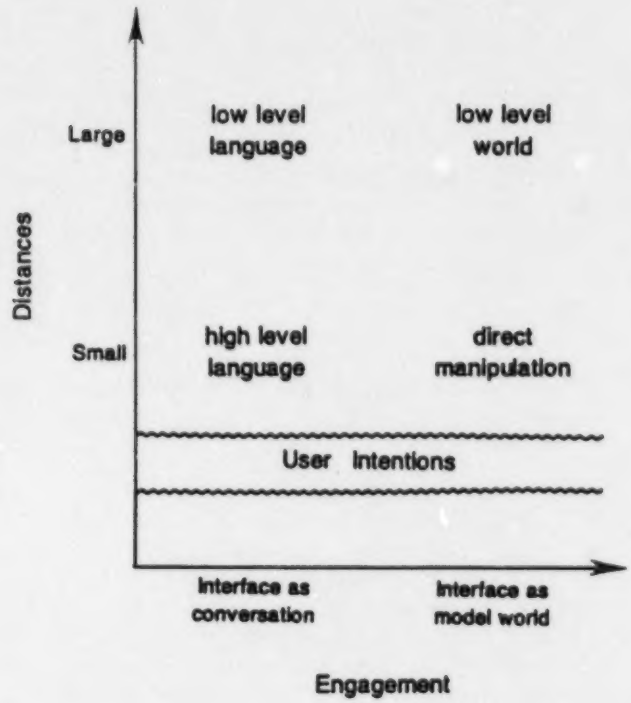

FIGURE 8
A space of interfaces. The dimensions of distance and engagement form a space of interfaces within which we can locate some familiar types of interfaces. Direct manipulation interfaces are those that minimize the distances and maximize engagement. As always, the distance between user intentions and the interface language depends upon the nature of the task the user is performing.

apt to provide the weakest semantic match between intentions and the language of the interface. In this case, the interface is an intermediary between the user and the task. Worse, it is an intermediary that does not understand actions at the level of description in which the user likes to think of them. Here the user must translate intentions into complex or lengthy expressions in the language that the interface intermediary can understand.

A more direct situation arises when the central metaphor of the interface is a world. Then the user can be directly engaged with the objects in a world, but still, if the actions in that world do not match those that the user wishes to perform within the task domain, getting the task done may be a difficult process. The user may believe that things are getting done and may even experience a sense of engagement with the world, yet still be doing things at too low a level. This is the state of some of the recently introduced direct manipulation systems: They produce an immediate sense of engagement, but as the user develops experience with the system, the interface appears clumsy, to interfere too much, too demand too many actions and decisions at the wrong level of specification. These interfaces appear on the surface to be Direct Manipulation interfaces, but they fail to produce the proper feelings of Direct Engagement with the task world. Beginners tend to like them: Opinions among experts vary.

Closing the distance between the user's intentions and the level of specification of the interface language allows the user to make efficient specifications of intentions. Where this is done with a high level language, quite efficient interfaces can be designed. An intermediary is provided that does just what the user wants done: no more, no less. This is the situation in

most modern integrated programming environments. For some classes of tasks, such interfaces may be superior to direct manipulation interfaces.

Finally, the most direct of the interfaces will lie where engagement is maximized, where just the correct semantic and articulatory matches are provided, and where all distances are minimized.

PROBLEMS WITH DIRECT MANIPULATION

Direct Manipulation systems have both virtues and vices. The immediacy of feedback and the natural translation of intentions to actions make some tasks easy. The matching of levels of thought to the interface language—semantic directness—increases the ease and power of performing some activities at a potential cost of generality and flexibility. Not all things should be done directly. Thus, a repetitive operation is probably best done via a script, that is, through a symbolic description of the tasks that are to be accomplished. Direct Manipulation interfaces have difficulty handling variables, or distinguishing the depiction of an individual element from a representation of a set or class of elements. Direct Manipulation interfaces have problems with accuracy, for the notion of mimetic action puts the responsibility on the user to control actions with precision, a responsibility that is often best handled through the intelligence of the system, and sometimes best communicated symbolically.

A more fundamental problem with direct manipulation interfaces arises from the fact that much of the appeal and power of this form of interface comes from their ability to directly support the way we normally think about a domain. They amplify our knowledge of the domain and allow us to think in the familiar terms of the application domain rather than those of the medium of computation. But if we restrict ourselves to only building interfaces that allow us to do things we can already do and to think in ways we already think, we will miss the most exciting potential of new technology: to provide new ways to think of and to interact with a domain. Providing these new ways and creating conditions that will make them feel direct and natural is an important challenge to the interface designer.

Direct manipulation interfaces are not a panacea. Although with sufficient practice many interfaces can come to feel direct, a properly designed interface, one which exploits semantic and articulatory directness, should decrease the amount of learning required and provide a natural mapping to the task. But interface design is subject to many tradeoffs. There are surely instances when one might wisely trade off directness for generality, or for more facile ways of saying abstract things. In interfaces that are not restricted to a single domain, it may be difficulty or even impossible to satisfy the requirements for directness. Some of the properties of Direct Manipulation systems are not wholly satisfactory in terms of the direction of the focus of attention and level of analysis. And the articulatory directness involved in pointing at objects might need to be traded off against the difficulties of moving the hands between input devices or of problems in pointing with great precision.

It is important not to equate directness with ease of use. Indeed, if the interface is really invisible, then the difficulties within the task domain get transferred directly into difficulties for the user. Suppose the user struggles to formulate an intention because of lack of knowledge of the task domain. The user may complain that the system is difficult to use. But the difficulty is in the task domain, not in the interface language. Direct Manipulation interfaces do not pretend to assist in overcoming problems that result from poor understanding of the task domain.

Reassessing the Claims for Direct Manipulation

What about the claims for direct manipulation? Do these systems really meet the hopes? Alas, it is too early to tell. We believe that direct manipulation systems carry gains in ease of learning and ease of use. If the mapping is done correctly, then both the form and the meaning of commands should be easier to acquire and retain. Interpretation of the output should be immediate and straight forward. If the interface is a model of the task domain, then one

could have the feeling of directly engaging the problem of interest itself. It is sometimes said that in such situations the interface disappears. It is probably more revealing to say that the interface is no longer recognized as an interface. Instead, it is taken to be the task domain itself.

But are these desirable features? Are the tradeoffs too costly? As always, we are sure that the answer will depend upon the tasks to be accomplished. Certain kinds of abstraction that are easy to deal with in language seem difficult in a concrete model of a task domain. When we give up the conversation metaphor, we also give up dealing in descriptions, and in some contexts, there is great power in descriptions. As an interface to a programming task, direct manipulation interfaces are problematic. We know of no really useful direct manipulation programming environments. Issues such as controlling the scope of variable bindings promise to be quite tricky in the direct manipulation environments. Will Direct Manipulation systems live up to their promise? Yes and no. Basically, the systems will be good and powerful for some purposes, poor and weak for others. In the end, many things done today will be replaced by Direct Manipulation systems. But we will still have conventional programming languages.

On the surface, the fundamental idea of a Direct Manipulation interface to a task flies in the face of two thousand years of development of abstract formalisms as a means of understanding and controlling the world. Until very recently, the use of computers has been an activity squarely in that tradition. So the exterior of Direct Manipulation, providing as it does for the direct control of a specific task world, seems somehow atavistic, a return to concrete thinking. On the inside, of course, the implementation of direct manipulation systems is yet another step in that long formal tradition. The illusion of the absolutely manipulable concrete world is made possible by the technology of abstraction.

Earlier in this chapter we reprinted two sets of descriptions of the virtues of direct manipulation systems, claims made in the early enthusiasm of their first discovery. Now that we have examined the many aspects of directness, it is time to go back and re-evaluate those claims. Let us see what we can say about them now. First, Shneiderman (1982) pointed at three "essential features" of Direct Manipulation. Let us examine each of the three features: Shneiderman's comments in *italics*, our assessment in regular font.

1. *Continuous representation of the object of interest.*

   This relates to the directness of evaluation, and as our analyses show, seems an essential aspect of a direct engagement system.

2. *Physical actions or labeled button presses instead of complex syntax.*

   The feature of "physical actions" we interpret to refer to articulatory directness, and more particularly, to what one might call "mimetic directness," actions that mimic the desired changes on the objects of interest, especially movement or size changes. "Labelled button presses" must refer to semantic directness. The former clearly is important in establishing direct engagement. The latter is not. The latter operation really seems irrelevant to arguments about this form of directness, whatever the virtues for ease of learning or ease of use.

3. *Rapid incremental reversible operations whose impact on the object of interest is immediately visible.*

   This is perhaps the essence of direct engagement: It reflects what we and Draper (in press) call the importance of "inter-referential I/O." The important point to us, implicitly stated in this assumption, is that the objects upon which the actions are taken are exactly the same as those upon which evaluation is made. The reversibility of the operations is desirable, but not necessary. Not all operations can have this feature in a natural way. So too with immediacy of the result: Where immediacy is a natural part of the domain, then this is essential. Otherwise, it is desirable, but may not always be necessary—as we have discussed at length within this chapter.

BEST COPY AVAILABLE

If we reinterpret Shneiderman's claims to be about *Direct Engagement*, then his three features fare well: They do seem to help define what is necessary to develop a direct engagement system. Now let us look at Shneiderman's six claims for the results of such a system:

1. *Novices can learn basic functionality quickly, usually through a demonstration by a more experienced user.*

   This may or may not be true. We think it really derives from the fact that a good Direct Manipulation interface is invisible—the user feels as if operations are actually done directly on the task domain. And if the computer novice is already knowledgeable in the task domain, then much of what is needed to use the interface is already known.

   Why might training be possible through demonstration? Well, two reasons. Lewis (in press) argues that demonstration is, in general, a superior method of instruction. Think of a demonstration as a dynamic example. In this case, then, the superiority of demonstration has nothing to do with direct manipulation. But there is a second reason. Owen (in press) points out that demonstrations with normal, command language systems, are often puzzling because the actions are not visible. That is, the expert waves hands over the keyboard and mysterious wonderful results appear on the screen. The learner often has little notion of what operation was performed. But with a typical direct manipulation (read direct engagement) system, the actions themselves are visible, and their results are both visible and also direct reflections of the operations done upon them. Whenever these cases apply, we believe the claim will be valid.

2. *Experts can work extremely rapidly to carry out a wide range of tasks, even defining new functions and features.*

   We are suspicious of this claim. In fact, we would not be surprised if experts are *slower* with Direct Manipulation systems than with command language systems. We suspect that the virtues of Direct Manipulation lie elsewhere: Speed at execution is not likely to be a relevant factor. Real experts can probably type a few lines of obscure code much faster than they could move objects around a screen, position them properly, and do the necessary pointing operations.

3. *Knowledgeable intermittent users can retain operational concepts.*

   This could be true, but if so it probably reflects two things: The expertise at usage really reflects expertise in the subject matter, which is probably well established and apt to fade slowly from memory, if at all; and a good semantic mapping of actions leads to slower forgetting and also easier rederivation of the operations that are forgotten. We suspect that these claims are true, but derive from aspects that would be true for many well-designed systems, not just Direct Manipulation systems.

4. *Error messages are rarely needed.*

   Yes and No. Error messages are often not needed because the results are immediately visible, and because some classes of errors are not even possible. But Direct Manipulation systems have their own problems. It is possible to make new classes of errors, some of them potentially serious. Worse, because these are apt to be errors in the task domain (but legal operations as far as the interface is concerned) they are hard to detect. Finally, Direct Manipulation systems sometimes simply don't bother with error messages, assuming that the ease of evaluation will make it obvious to the user that the desired operation was not done. It's not that the message wasn't needed, it's just that the system didn't bother to present one. This what Lewis and Norman call the "do nothing" strategy. This strategy is not always desirable (see Lewis & Norman, in press).

5. *Users can see immediately if their actions are furthering their goals, and if not, they can simply change the direction of their activity.*

   The first part of the claim results from the properties of Direct Evaluation. The second part has already been discussed in our response to feature 3 of the first list. But basically, the ability to "change the direction of their activity" results from the natural reversibility of many actions. For those actions that are not so naturally reversible, the systems do not fare any differently than more conventional systems. Immediate feedback as to the outcome of an operation and "undo" commands are valuable for any system, not just Direct Manipulation systems.

6. *Users have reduced anxiety because the system is comprehensible and because actions are so easily reversible.*

   This is outside the domain of our analyses and difficult to assess. A fair comparison would require two systems, both equally well matched in capability and in semantic directness.

In conclusion, some of the claims seem too strong, some represent features of many well-designed systems, not just direct manipulation systems, and some seem to be correct. All in all, the early claims seem to fare reasonably well, although today we can bring more sophistication and depth of analysis to bear upon them then was possible at the time they were made.

CONCLUDING REMARKS

Direct Manipulation implies a directness of action, a directness of the translation of intentions to actions, and a directness in the feedback and knowledge of the system. The feeling of directness also implies a feeling of control over the objects in the task domain—a Direct Engagement which results, in part, from the feeling that one is operating directly upon the objects, where the objects upon which actions are performed are the very same as those from which output is received.

In this chapter we have concentrated upon the various aspects of directness with respect to the performance of a task, with little mention of the problems of learning. Nonetheless, a system that is properly designed should also be one that is easy to learn. Our discussions of the nature of the Gulfs separating a person's intentions and evaluations from the system's inputs and outputs provide a foundation for making the learning task easier. If the Gulfs of Execution and Evaluation are properly bridged from the system side, then the system should be relatively easy to learn: Minimum processing is required to get from intention to action and from outcome to interpretation. The learning required should be concentrated in the task domain. The hope is that people will be able to spend their time learning the task domain, not learning the computer system. Indeed, a properly developed Direct Manipulation interface should appear to the user as if it is the task that is being executed directly: The computer system and its interface will be more or less invisible. Much empirical work remains to be done to substantiate these claims.

The understanding of Direct Manipulation interfaces is complex. There are a host of virtues and vices. Do not be dismayed by the complexity of the arguments. To our mind, direct manipulation interfaces provide exciting new perspectives on possible modes of interactions with computers. It is our intention to explore these modes further, through the building of working systems, expanding the theoretical analysis, and performing empirical evaluations. We believe that this direction of work has the potential to deliver important new conceptualizations and a new philosophy of interaction.

ACKNOWLEDGMENTS

We thank Ben Shneiderman for his helpful comments on an earlier draft of the chapter, Eileen Conway for her aid with the illustrations, and Julie Norman and Sondra Buffett for extensive editorial comments.

REFERENCES

Borning, A. (1979). *Thinglab: A constraint-oriented simulation laboratory* (Tech. Rep. SSL-79-3). Palo Alto: Xerox PARC.

Budge, B. (1983). *Pinball construction set* [Computer program]. San Mateo, CA: Electronic Arts.

Buxton, W. (in press). There's more to interaction that meets the eye: Some issues in manual input. In D. A. Norman & S. W. Draper (Eds.), *User centered system design: New perspectives in human-machine interaction.* Hillsdale, NJ: Erlbaum.

Draper, S. W. (in press). Display managers as the basis for user-machine communication. In D. A. Norman & S. W. Draper (Eds.), *User centered system design: New perspectives in human-machine interaction.* Hillsdale, NJ: Erlbaum.

Galton, F. (1894). Arithmetic by smell. *Psychological Review, 1,* 61-62.

Hollan, J. D., Hutchins, E., & Weitzman, L. (1984). STEAMER: An interactive inspectable simulation-based training system. *AI Magazine, 5,* 15-27.

Hollan, J. D., Stevens, A., & Williams, M. D. (1980). STEAMER: An advanced computer-assisted instruction system for propulsion engineering. *Proceedings of Summer Computer Simulation Conference,* 400-404.

Kay, A. (1984, March). Computer software. *Scientific American,* pp. 52-59.

Laurel, B. K. (in press). Interface as mimesis. In D. A. Norman & S. W. Draper (Eds.), *User centered system design: New perspectives in human-machine interaction.* Hillsdale, NJ: Erlbaum.

Lewis, C. (in press). Understanding what's happening in system interactions. In D. A. Norman & S. W. Draper (Eds.), *User centered system design: New perspectives in human-machine interaction.* Hillsdale, NJ: Erlbaum.

Lewis, C., & Norman, D. A. (in press). Designing for error. In D. A. Norman & S. W. Draper (Eds.), *User centered system design: New perspectives in human-machine interaction.* Hillsdale, NJ: Erlbaum.

Minsky, M. (1967). Why programming is a good medium for expressing poorly-understood and sloppily-formulated ideas. In M. Krampen & P. Seitz (Eds.), *Design and planning II: Computers in design and communication* (pp. 117-121). New York: Visual Committee Books, Hastings House.

Norman, D. A. (in press). Cognitive engineering. In D. A. Norman & S. W. Draper (Eds.), *User centered system design: New perspectives in human-machine interaction.* Hillsdale, NJ: Erlbaum.

Owen, D. (in press). Answers first, the questions. In D. A. Norman & S. W. Draper (Eds.),

User centered system design: New perspectives in human-machine interaction. Hillsdale, NJ: Erlbaum.

Perlis, A. J. (1982). Epigrams on programming. *SIGPLAN Notices, 17*(9), 7-13. Riley, M. S. (in press). User understanding. In D. A. Norman & S. W. Draper (Eds.), *User centered system design: New perspectives in human-machine interaction.* Hillsdale, NJ: Erlbaum.

Riley, M., & O'Malley, C. (1984). Planning nets: A framework for analyzing user-computer interactions. In B. Shackel (Ed.), *INTERACT '84: First Conference on Human-Computer Interaction.* Amsterdam: North-Holland.

Shneiderman, B. (1974). A computer graphics system for polynomials. *The Mathematics Teacher, 67*(2), 111-113.

Shneiderman, B. (1982). The future of interactive systems and the emergence of direct manipulation. *Behavior and Information Technology, 1*, 237-256.

Shneiderman, B. (1983). Direct manipulation: A step beyond programming languages. *IEEE Computer, 16*(8), 57-69.

Sutherland, I. E. (1963). Sketchpad: A man-machine graphical communication system. *Proceedings of the Spring Joint Computer Conference*, 329-346.

ICS Technical Report List

The following is a list of publications by people in the Institute for Cognitive Science. For reprints, write or call:

Institute for Cognitive Science, C-015
University of California, San Diego
La Jolla, CA 92093
(619) 452-6771

8301. David Zipser. *The Representation of Location.* May 1983.

8302. Jeffrey Elman and Jay McClelland. *Speech Perception as a Cognitive Process: The Interactive Activation Model.* April 1983. Also published in N. Lass (Ed.), *Speech and language: Volume 10*, New York: Academic Press, 1983.

8303. Ron Williams. *Unit Activation Rules for Cognitive Networks.* November 1983.

8304. David Zipser. *The Representation of Maps.* November 1983.

8305. The HMI Project. *User Centered System Design: Part I, Papers for the CHI '83 Conference on Human Factors in Computer Systems.* November 1983. Also published in A. Janda (Ed.), *Proceedings of the CHI '83 Conference on Human Factors in Computing Systems.* New York: ACM, 1983.

8306. Paul Smolensky. *Harmony Theory: A Mathematical Framework for Stochastic Parallel Processing.* December 1983. Also published in *Proceedings of the National Conference on Artificial Intelligence, AAAI-83*, Washington DC, 1983.

8401. Stephen W. Draper and Donald A. Norman. *Software Engineering for User Interfaces.* January 1984. Also published in *Proceedings of the Seventh International Conference on Software Engineering*, Orlando, FL, 1984.

8402. The UCSD HMI Project. *User Centered System Design: Part II, Collected Papers.* March 1984. Also published individually as follows: Norman, D.A. (in press), Stages and levels in human-machine interaction, *International Journal of Man-Machine Studies*; Draper, S.W., The nature of expertise in UNIX; Owen, D., Users in the real world; O'Malley, C., Draper, S.W., & Riley, M., Constructive interaction: A method for studying user-computer-user interaction; Smolensky, P., Monty, M.L., & Conway, E., Formalizing task descriptions for command specification and documentation; Bannon, L.J., & O'Malley, C., Problems in evaluation of human-computer

interfaces: A case study; Riley, M., & O'Malley, C., Planning nets: A framework for analyzing user-computer interactions; all published in B. Shackel (Ed.), *INTERACT '84, First Conference on Human-Computer Interaction*, Amsterdam: North-Holland, 1984; Norman, D.A., & Draper, S.W., Software engineering for user interfaces, *Proceedings of the Seventh International Conference on Software Engineering*, Orlando, FL, 1984.

8403. Steven L. Greenspan and Eric M. Segal. *Reference Comprehension: A Topic-Comment Analysis of Sentence-Picture Verification.* April 1984. Also published in *Cognitive Psychology, 16*, 556-606, 1984.

8404. Paul Smolensky and Mary S. Riley. *Harmony Theory: Problem Solving, Parallel Cognitive Models, and Thermal Physics.* April 1984. The first two papers are published in *Proceedings of the Sixth Annual Meeting of the Cognitive Science Society*, Boulder, CO, 1984.

8405. David Zipser. *A Computational Model of Hippocampus Place-Fields.* April 1984.

8406. Michael C. Mozer. *Inductive Information Retrieval Using Parallel Distributed Computation.* May 1984.

8407. David E. Rumelhart and David Zipser. *Feature Discovery by Competitive Learning.* July 1984. Also published in *Cognitive Science, 9*, 75-112, 1985.

8408. David Zipser. *A Theoretical Model of Hippocampal Learning During Classical Conditioning.* December 1984.

8501. Ronald J. Williams. *Feature Discovery Through Error-Correction Learning.* May 1985.

8502. Ronald J. Williams. *Inference of Spatial Relations by Self-Organizing Networks.* May 1985.

8503. Edwin L. Hutchins, James D. Hollan, and Donald A. Norman. *Direct Manipulation Interfaces.* May 1985. To be published in D. A. Norman & S. W. Draper (Eds.), *User Centered System Design: New Perspectives in Human-Computer Interaction.* Hillsdale, NJ: Erlbaum.

Earlier Reports by People in the Cognitive Science Lab

The following is a list of publications by people in the Cognitive Science Lab and the Institute for Cognitive Science. For reprints, write or call:

Institute for Cognitive Science, C-015
University of California, San Diego
La Jolla, CA 92093
(619) 452-6771

ONR-8001. Donald R. Gentner, Jonathan Grudin, and Eileen Conway. *Finger Movements in Transcription Typing*. May 1980.

ONR-8002. James L. McClelland and David E. Rumelhart. *An Interactive Activation Model of the Effect of Context in Perception: Part I*. May 1980. Also published in *Psychological Review, 88,5*, pp. 375-401, 1981.

ONR-8003. David E. Rumelhart and James L. McClelland. *An Interactive Activation Model of the Effect of Context in Perception: Part II*. July 1980. Also published in *Psychological Review, 89, 1*, pp. 60-94, 1982.

ONR-8004. Donald A. Norman. *Errors in Human Performance*. August 1980.

ONR-8005. David E. Rumelhart and Donald A. Norman. *Analogical Processes in Learning*. September 1980. Also published in J. R. Anderson (Ed.), *Cognitive skills and their acquisition*. Hillsdale, NJ: Erlbaum, 1981.

ONR-8006. Donald A. Norman and Tim Shallice. *Attention to Action: Willed and Automatic Control of Behavior*. December 1980.

ONR-8101. David E. Rumelhart. *Understanding Understanding*. January 1981.

ONR-8102. David E. Rumelhart and Donald A. Norman. *Simulating a Skilled Typist: A Study of Skilled Cognitive-Motor Performance*. May 1981. Also published in *Cognitive Science, 6*, pp. 1-36, 1982.

ONR-8103. Donald R. Gentner. *Skilled Finger Movements in Typing*. July 1981.

ONR-8104. Michael I. Jordan. *The Timing of Endpoints in Movement*. November 1981.

ONR-8105. Gary Perlman. *Two Papers in Cognitive Engineering: The Design of an Interface to a Programming System and MENUNIX: A Menu-Based Interface to UNIX (User Manual)*. November 1981. Also published in *Proceedings of the 1982 USENIX Conference*, San Diego, CA, 1982.

ONR-8106. Donald A. Norman and Diane Fisher. *Why Alphabetic Keyboards Are Not Easy to Use: Keyboard Layout Doesn't Much Matter*. November 1981. Also published in *Human Factors, 24*, pp. 509-515, 1982.

ONR-8107. Donald R. Gentner. *Evidence Against a Central Control Model of Timing in Typing*. December 1981. Also published in *Journal of Experimental Psychology: Human Perception and Performance, 8*, pp. 793-810, 1982.

ONR-8201. Jonathan T. Grudin and Serge Larochelle. *Digraph Frequency Effects in Skilled Typing*. February 1982.

ONR-8202. Jonathan T. Grudin. *Central Control of Timing in Skilled Typing*. February 1982.

ONR-8203. Amy Geoffroy and Donald A. Norman. *Ease of Tapping the Fingers in a Sequence Depends on the Mental Encoding*. March 1982.

ONR-8204. LNR Research Group. *Studies of Typing from the LNR Research Group: The role of context, differences in skill level, errors, hand movements, and a computer simulation.* May 1982. Also published in W. E. Cooper (Ed.), *Cognitive aspects of skilled typewriting*. New York: Springer-Verlag, 1983.

ONR-8205. Donald A. Norman. *Five Papers on Human-Machine Interaction.* May 1982. Also published individually as follows: Some observations on mental models, in D. Gentner and A. Stevens (Eds.), *Mental models*, Hillsdale, NJ: Erlbaum, 1983; A psychologist views human processing: Human errors and other phenomena suggest processing mechanisms, in *Proceedings of the International Joint Conference on Artificial Intelligence*, Vancouver, 1981; Steps toward a cognitive engineering: Design rules based on analyses of human error, in *Proceedings of the Conference on Human Factors in Computer Systems*, Gaithersburg, MD, 1982; The trouble with UNIX, in *Datamation*, *27*,12, November 1981, pp. 139-150; The trouble with networks, in *Datamation*, January 1982, pp. 188-192.

ONR-8206. Naomi Miyake. *Constructive Interaction.* June 1982.

ONR-8207. Donald R. Gentner. *The Development of Typewriting Skill.* September 1982. Also published as Acquisition of typewriting skill, in *Acta Psychologica, 54*, pp. 233-248, 1983.

ONR-8208. Gary Perlman. *Natural Artificial Languages: Low-Level Processes.* December 1982. Also published in *The International Journal of Man-Machine Studies, 20*, pp. 373-419, 1984.

ONR-8301. Michael C. Mozer. *Letter Migration in Word Perception.* April 1983. Also published in *Journal of Experimental Psychology: Human Perception and Performance, 9, 4*, pp. 531-546, 1983.

ONR-8302. David E. Rumelhart and Donald A. Norman. *Representation in Memory*. June 1983. To appear in R. C. Atkinson, G. Lindzey, & R. D. Luce (Eds.), *Handbook of experimental psychology*. New York: Wiley (in press).